

# Constraint Programming Models for Graceful Graphs

Barbara M. Smith

Cork Constraint Computation Centre, University College Cork, Ireland  
b.smith@4c.ucc.ie

**Abstract.** The problem of finding a graceful labelling of a graph, or proving that the graph is not graceful, has previously been modelled as a CSP. A new and much faster CSP model of the problem is presented, with several new results for graphs whose gracefulness was previously unknown. Several classes of graph that are conjectured to be graceful only for small instances are investigated: after a certain size, it appears that for some of these classes the search to prove that there is no graceful labelling is essentially the same for each successive instance. The possibility of constructing a proof of the conjecture based on the search is discussed.

## 1 Introduction

The study of graceful labellings of graphs is an active research area in graph theory. Few general results are known and constraint programming can be a useful tool in investigating new classes of graph. An existing CSP model used in previous studies is shown to be far slower than a model derived from a proof that cliques  $K_n$  are not graceful for  $n > 4$ . The new model is applied to instances from several classes of graph, finding some new graceful labellings and showing that some graphs, whose status was previously unknown, are not graceful. For some classes of graph, the results lead to the conjecture that only small instances of the class are graceful. The paper discusses whether it would be possible to base a proof of such a conjecture on a trace of the search to show that an instance of the class is not graceful.

## 2 Graceful Graphs

A labelling  $f$  of the nodes of a graph with  $q$  edges is *graceful* if  $f$  assigns each node a unique label from  $\{0, 1, \dots, q\}$  and when each edge  $xy$  is labelled with  $|f(x) - f(y)|$ , the edge labels are all different. (Hence, the edge labels are a permutation of  $1, 2, \dots, q$ .)

Figure 1 shows an example, with 25 edges and 10 nodes. The labels on the edges consist of the numbers 1 to 25, while the node labels are all different and taken from  $\{0, \dots, 25\}$ .

Gallian [6] gives a survey of graceful graphs, i.e. graphs with a graceful labelling, and lists the graphs whose status is known; his survey is frequently updated to include new results. Graceful labellings were first defined by Rosa in 1967, although the name was introduced by Golomb [8] in 1972. Gallian lists a number of applications of labelled graphs; however, the study of graceful graphs has become an active area of research in

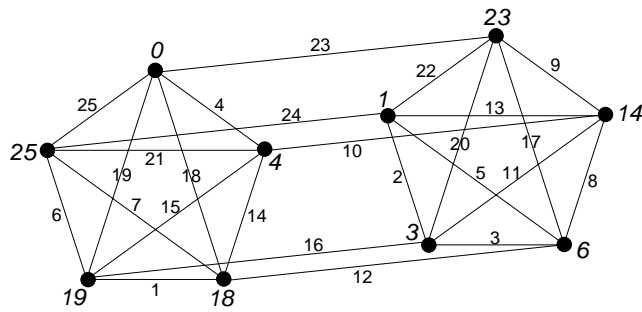


Fig. 1. A graceful labelling of the graph  $K_5 \times P_2$ .

its own right. The survey lists several classes of graph for which it is known that every instance is graceful, for instance the wheel graph  $W_n$ , consisting of a cycle  $C_n$  and an additional node joined to every node of the cycle. However, the only general result given by Gallian is that if every node has even degree and the number of edges is congruent to 1 or 2 (mod 4) then the graph is not graceful. For example, the cycles  $C_{4n+1}$  and  $C_{4n+2}$  are not graceful, although  $C_{4n}$  and  $C_{4n+3}$  are graceful for all  $n$ . There is a long-standing conjecture that all trees are graceful and although this has been proved for several classes of tree (including paths), and for all trees with at most 27 nodes, the general case remains unproven.

Given a graph whose gracefulness is so far unknown, in general there is no way to tell whether it is graceful or not, except by trying to label it. Constraint programming is thus a useful tool to use in investigating graceful graphs.

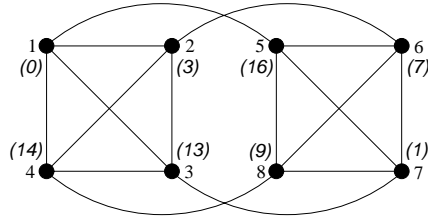
### 3 A CP Model

Constraint programming has already been applied to finding graceful labellings in a few cases; for instance, the all-interval series problem (problem 007 in CSPLib, at <http://www.csplib.org>) is equivalent to finding a graceful labelling of a path. Lustig & Puget [10] found a graceful labelling of the graph shown in Figure 2, which was not previously known to be graceful. Petrie and Smith [11] used graceful graphs to investigate different symmetry breaking methods in constraint programming.

In order to model the problem of finding a graceful labelling as a CSP, the nodes of the graph are numbered from 1 to  $n$ , where  $n$  is the number of nodes. Figure 2 shows this numbering of the nodes, as well as (in brackets) the label attached to each node in one of the graceful labellings.

The CSP model used in [10, 11] has two sets of variables: a variable for each node,  $x_1, x_2, \dots, x_n$  each with domain  $\{0, 1, \dots, q\}$  and a variable for each edge,  $d_1, d_2, \dots, d_q$ , each with domain  $\{1, 2, \dots, q\}$ . The value assigned to  $x_i$  is the label attached to node  $i$ , and the value of  $d_k$  is the label attached to the edge  $k$ .

The constraints of the problem are: if edge  $k$  joins nodes  $i$  and  $j$  then  $d_k = |x_i - x_j|$ , for  $k = 1, 2, \dots, q$ ;  $x_1, x_2, \dots, x_n$  are all different;  $d_1, d_2, \dots, d_q$  are all different (and form a permutation).



**Fig. 2.** The graph  $K_4 \times P_2$ .

There are two kinds of symmetry in the problem of finding a graceful labelling of a graph: first, there may be symmetry in the graph. For instance, if the graph is a clique, any permutation of the node labels in a graceful labelling is also graceful, and if the graph is a path,  $P_n$ , the node labels can be reversed. The second type of symmetry is that we can replace the value  $v$  of every node variable  $x_i$  by its complement  $q - v$ . We can also combine each graph symmetry with the complement symmetry. For instance, the graceful labelling  $(0, 3, 1, 2)$  of  $P_4$  has three symmetric equivalents: reversal  $(2, 1, 3, 0)$ ; complement  $(3, 0, 2, 1)$ ; and reversal + complement  $(1, 2, 0, 3)$ .

If the graph is symmetric, it is essential to eliminate all or most of the symmetry in the CSP in order to avoid wasted search, especially if all graceful labellings are required or the graph is not graceful. In terms of the CSP model described above, the symmetry of the graph leads to variable symmetries in the CSP affecting the node variables  $x_1, x_2, \dots, x_n$ ; each symmetry maps an assignment of a value  $l$  to a variable  $x_i$  to an assignment of the same value to a variable  $x_j$ . Following the procedure introduced by Crawford *et al.* [5], variable symmetries can be eliminated by adding a lexicographic ordering constraint for each symmetry. Although in general this may not be practicable, Puget [12] showed that when there is an allDifferent constraint on the variables, the symmetry can be eliminated by at most  $n - 1$  binary constraints; he used the CSP model of graceful graphs described in this section as an example.

The symmetries of the graph in Figure 2 are, firstly, any permutation of the 4-cliques which acts on both in the same way, for instance, transposing nodes 1 and 2 and simultaneously nodes 5 and 6. Secondly, the labels of the first clique (nodes 1, 2, 3, 4) can be interchanged with the labels of the corresponding nodes (nodes 5, 6, 7, 8) in the second. A possible set of constraints to eliminate the graph symmetry is  $x_1 < x_2, x_2 < x_3, x_3 < x_4$  to exclude permutations within the cliques and  $x_1 < x_5, x_1 < x_6, x_1 < x_7, x_1 < x_8$  to exclude swapping the cliques and permuting both. Since the constraints imply that  $x_1 = 0$ , this constraint together with  $x_2 < x_3, x_3 < x_4$  is sufficient.

It is not easy to devise a simple constraint that will eliminate the complement symmetry in this example. It is simpler if the graph has a node that is not symmetrically equivalent to any other node; the constraint that the label of this node must be  $\leq q/2$  will eliminate the complement symmetry, at least if  $q$  is odd. Eliminating the complement symmetry will be returned to later; for now, this symmetry will be ignored.

As an alternative to using symmetry-breaking constraints, a dynamic symmetry-breaking method such as Symmetry Breaking During Search (SBDS) [7] can be used; again, this will be discussed further in the context of an alternative CSP model.

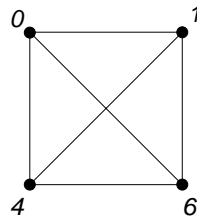
The model just described, which will be called the *node model* in the rest of the paper, is adequate to find all graceful labellings of the graph in Figure 2, and allowed us to investigate slightly larger graphs as well, e.g. the related graphs  $K_5 \times P_2$  and  $K_6 \times P_2$ .  $K_5 \times P_2$  is graceful, and has a unique graceful labelling, apart from symmetric equivalents, shown in Figure 1, whereas  $K_6 \times P_2$  is not graceful; these new results are now listed in Gallian’s survey [6]. Results that we obtained with this model for other classes of graph are summarised in [13]. However, the search effort and run time increase rapidly with problem size, so that it is very time-consuming, as shown below, to prove that  $K_7 \times P_2$ , with 14 nodes and 49 edges, is not graceful;  $K_8 \times P_2$  is out of reach within a reasonable time. In the next sections, a better model of the problem is introduced.

## 4 Gracefulness of Cliques

Beutner & Harborth [3] present a search algorithm which they use to investigate graceful labellings of “nearly complete” graphs. To illustrate their method, they present Golomb’s proof that  $K_n$  is not graceful for  $n > 4$  [8]:

Any gracefully labelled graph must have an edge labelled  $q$  and it must link nodes labelled 0 and  $q$ . Then we consider the edge labelled  $q - 1$  (which has to exist unless the graph is just a path of length 2). It must link either 0 and  $q - 1$  or 1 and  $q$ . So we must have a path  $0 \leftrightarrow q \leftrightarrow 1$  or  $q - 1 \leftrightarrow 0 \leftrightarrow q$ . These two possibilities are symmetrically equivalent under the complement symmetry, so we can break this symmetry by arbitrarily choosing one, say  $0 \leftrightarrow q \leftrightarrow 1$ ; hence, we need a third node labelled 1.

This gives a complete graceful labelling of  $K_3$ . To extend the labelling to  $K_n$ ,  $n > 3$ , we next consider the edge labelled  $q - 2$ . This must link either 0 and  $q - 2$  or 1 and  $q - 1$  or 2 and  $q$ , so we need a new node labelled  $q - 2$ ,  $q - 1$  or 2. But there is already an edge labelled 1 (from 0 to 1) and both  $q - 1$  (from  $q - 1$  to  $q$ ) and 2 (from 2 to 1) give another, which is not allowed. Hence, there must be nodes labelled 0, 1,  $q - 2$  and  $q$ . This gives the graceful labelling of  $K_4$ , with  $q = 6$ , shown in Figure 3; the construction shows that it is unique, apart from the complement labelling.



**Fig. 3.** The graceful labelling of  $K_4$

In larger cliques, the next edge label to consider is  $q - 4$  (there is already an edge labelled  $q - 3$  from 1 to  $q - 2$ ). The fifth node added to the clique must be labelled so that this edge label is created. The possible ways to construct an edge label  $q - 4$  (in

any graph) are: 0 and  $q - 4$  or 1 and  $q - 3$  or 2 and  $q - 2$  or 3 and  $q - 1$  or 4 and  $q$ . However, labelling the fifth node with any of  $q - 4$ , 2 and 3 gives a new edge labelled 2 ( $q - 4$  to  $q - 2$ , 2 to 0, 3 to 1), so the fifth node must be labelled 4. But this gives an edge labelled  $q - 6$  from  $q - 2$  to 4; we also have an edge labelled 4, from 0 to 4, and for  $n = 5$ ,  $q - 6 = 4$ . So  $K_5$  cannot be graceful.

For  $n \geq 6$ , introducing a fifth node labelled 4 is allowable, so that the nodes so far are labelled 0, 1, 4,  $q - 2$  and  $q$ . The largest edge label not already present is  $q - 5$ . This edge label must come from nodes labelled *either* 0 and  $q - 5$  or 1 and  $q - 4$  or 2 and  $q - 3$  or 3 and  $q - 2$  or 4 and  $q - 1$  or 5 and  $q$ . Nodes labelled  $q - 4$ , 2 and 3 have already been ruled out.  $q - 5$  will give a 2nd edge labelled 3 (from  $q - 5$  to  $q - 2$ ); 5 will give a 2nd edge labelled 4, from 5 to 1. So we cannot have an edge labelled  $q - 6$  and hence  $K_n$  is not graceful for  $n > 5$ .

Beutner & Harborth use a backtracking search procedure, based on the approach in the proof, to find graceful labellings or prove that there are none. They focus on the edge labels, considering each label in turn, from  $q$  (the number of edges) downwards. Each edge label has to be used, and appear somewhere in the graph, and they construct a labelling, or prove that there is no labelling, by trying to decide, for each edge label in turn, which node labels will appear at either end of the edge. The classes they consider are nearly complete graphs, with specific kinds of subgraph removed from a clique; for instance, they show that  $K_n - e$ , i.e. an  $n$ -clique with one edge removed, is graceful only for  $n \leq 5$  and that any graph derived from  $K_n$  by removing 2 or 3 edges is not graceful for  $n > 6$ .

## 5 A CSP Model Based on Edge Labels

The proof that  $K_n$  is not graceful for  $n \geq 5$  can equally be adapted to modelling the problem of finding graceful labellings of a graph as a CSP. The model has a variable  $e_i$  for each edge label  $i$ , which indicates how this edge label will be formed. The value assigned to  $e_i$  is the smaller node label forming the edge labelled  $i$ , i.e. if  $e_i = j$ , the edge labelled  $i$  joins the nodes labelled  $j$  and  $j + i$ . Hence, the domain of  $e_i$  is  $\{0, 1, \dots, q - i\}$ . (Essentially, an edge label is associated with a *pair* of node labels, but since if the edge label and the smaller node label are known, the other node label is also known, we need only associate an edge label with one node label.) Note that the domain of  $e_q$  is  $\{0\}$ . The domain of  $e_{q-1}$  is initially  $\{0, 1\}$  but this can be reduced either to  $\{1\}$  or to  $\{0\}$  arbitrarily, to break the complement symmetry, as in the proof.

Once it is decided how to construct an edge label from two node labels, those node labels must be attached to specific nodes in the graph. (This is not necessary in the proof, because all nodes in a clique are equivalent.) The variable  $l_j$  represents the node to which the label  $j$  is attached, provided that this node label is used at all in the labelling. To allow for the fact that a label may not be used, the domains of these variables contain a dummy value, say  $n + 1$ .

The link between the two sets of variables is achieved by the constraints:  $e_i = j$  iff the values of  $l_j$  and  $l_{j+i}$  correspond to adjacent nodes ( $1 \leq i \leq q$ ;  $0 \leq j \leq q - i$ ). In Solver 6.0, this is expressed as a table constraint, using a ternary predicate which uses the adjacency matrix of the graph.

The node label variables must all have different values (unless they are assigned the dummy value). There is no need for a constraint to ensure that each edge has a different label, since an edge label is represented by a variable and so must have exactly one value, i.e. labels exactly one edge.

This new model will be called the *edge-label* model in the rest of the paper.

The complement symmetry can be eliminated by setting  $e_{q-1} = 1$ , as already described. (The alternative choice,  $e_{q-1} = 0$ , makes little difference.) However, the graph symmetry, if any, has still to be dealt with. In the edge-label model, it appears as value symmetries affecting the node-label variables. If a graph symmetry  $\sigma$  maps the node numbered  $i$  to the node numbered  $\sigma(i)$ , then it maps the assignment  $l_j = i$  to the assignment  $l_j = \sigma(i)$ , whereas in the node model, it maps  $x_i = j$  into  $x_{\sigma(i)} = j$ . Law and Lee [9] point out that value symmetry can be transformed into variable symmetry acting on the dual variables and then is amenable to the Crawford *et al.* procedure for deriving symmetry-breaking constraints.

Hence, a simple way to derive symmetry-breaking constraints in the edge-label model is to re-introduce the variables  $x_1, \dots, x_n$  from the node model. These are the duals of the node label variables  $l_1, \dots, l_q$ . The two sets of variables can be linked by the channelling constraints:  $x_i = j$  iff  $l_j = i$  for  $i = 1, \dots, n$  and  $j = 0, \dots, q$ , or equivalently by the global *inverse* constraint [1]. As a side-effect, the channelling constraints are sufficient to ensure that every possible node label is assigned to a different node, or else is not used, so that no explicit `allDifferent` constraint is required.

In the next section, where the two CSP models are compared experimentally, SBDS is also used to break the symmetry in the edge-label model; for some classes of graph, this is a better choice.

The search strategy is designed to mimic the process described in the proof of section 4, with an extra step in which specific nodes in the graph are labelled. The edge labels are considered in descending order; for each one in turn, the search first decides which node labels will make that edge label, and then decides where in the graph to put those node labels. Hence, both the edge label variables  $e_1, e_2, \dots, e_q$  and the node label variables  $l_0, l_1, \dots, l_q$  are search variables. The variable ordering strategy finds the largest edge label  $i$  that has not yet been attached to a specific edge in the graph. The next variable assigned is  $e_i$ , if that has not been assigned; if  $e_i$  has been assigned a value  $j$ , then the next variable is  $l_j$ , if that is not yet assigned, or  $l_{i+j}$ . (If all three variables have been assigned, the label  $i$  has been associated with a specific edge.)

In the next section, the performance of the edge-label model will be compared with that of the node model.

## 6 $K_n \times P_2$ Graphs

In this section, an experimental comparison of the two models is presented, using graphs of the class  $K_n \times P_2$ ; these have two copies of a clique  $K_n$  with corresponding nodes of successive cliques also forming the nodes of a path  $P_2$ . The graph  $K_4 \times P_2$  from this class was used in section 3 to illustrate the node model.

As before, the graph symmetry in the node model can be eliminated by the constraints  $x_1 = 0; x_2 < x_3 < \dots < x_n$ . The complement symmetry can now be broken

in the same way as in the edge-label model, by using the fact that the edge label  $q - 1$  either joins nodes labelled 1 and  $q$  or nodes 0 and  $q - 1$  and that these are symmetric under the complement symmetry. In the  $K_n \times P_2$  graphs, it is simple to add a constraint that there is a node labelled  $q - 1$  adjacent to the node labelled 0, since the other symmetry-breaking constraints ensure that node 1 is the node labelled 0.

The allDifferent constraint on the node variables is treated as a set of binary  $\neq$  constraints, whereas generalized arc consistency is enforced on the allDifferent constraint on the edge variables. They are treated differently because the values assigned to the edge variables form a permutation and hence the allDifferent constraint is much tighter than that on the node variables, which gives little scope for domain pruning.

In order to give a small search tree (even at the expense of increasing the run-time) generalized arc consistency is enforced on the ternary constraints  $d_k = |x_i - x_j|$ . A set of implied constraints are also added: for any triple of nodes  $i, j, k$ , where  $i$  is adjacent to both  $j$  and  $k$ ,  $2x_i \neq x_j + x_k$ .

Smallest-domain variable ordering is used with the node model for this class of graph; the earlier studies [11] showed that this is not always a good choice for these problems, compared to lexicographic variable ordering, but in this case it is better. In both models, the value ordering chooses the smallest available value in the domain.

In the edge-label model, as described in the last section, the graph symmetry can be broken by the symmetry-breaking constraints on the node variables  $x_1, x_2, \dots, x_n$ . For this class of problems, as will be seen, it is also worthwhile to use SBDS: the symmetry permuting the nodes within each cliques can be broken using SBDS and the symmetry between cliques by the constraint that the smallest node label in the first clique is less than the smallest in the second clique, i.e.  $\min(x_1, x_2, \dots, x_n) < \min(x_{n+1}, x_{n+2}, \dots, x_{2n})$ . This constraint still allows the nodes within the cliques to be freely permuted and so is compatible with SBDS breaking the symmetry within cliques. This symmetry is easy to deal with in SBDS: since it allows all permutations of the nodes in a clique (while simultaneously permuting the corresponding nodes in the other clique), it is only necessary to input the transpositions of pairs of nodes. For instance, one such transposition swaps the assignments to  $x_i$  and  $x_j$  ( $1 \leq i, j \leq n$ ) and also to  $x_{i+n}$  and  $x_{j+n}$ .

The models were implemented in ILOG Solver 6.0, and all experiments reported in this paper were run on a 1.7GHz Pentium M PC. The results for the two models, and both symmetry-breaking methods used in the edge-label model, are given in Table 1.

Clearly, the new model is far better than the original model for these problems. With hindsight, the domains of the node variables in the original model are too large to allow efficient search (and they increase rapidly with the size of the problem). In the edge-label model, the initial domain sizes of the edge-label variables, in the order in which they are assigned, are always  $1, 2, \dots, q$ . In practice, domain pruning ensures that by the time an edge label variable is reached during the search, there are very few possible values left in its domain – often no more than two.

For this class of graph, symmetry breaking constraints give significantly worse performance than SBDS, with the edge-label model. It is known that symmetry-breaking constraints can conflict with variable ordering heuristics. In the edge-label model, the symmetry-breaking constraints are posted on the node variables, but these are not search

**Table 1.** Comparison of CSP models for finding all graceful labellings of  $K_n \times P_2$  graphs, using Solver 6.0 on a 1.7GHz Pentium M PC.

Graph	Solutions	Node model (with symmetry-breaking constraints)		Edge label model			
		backtracks	time (sec.)	Symmetry-breaking constraints backtracks	time (sec.)	Breaking symmetry using SBDS backtracks	time (sec.)
$K_3 \times P_2$	4	37	0.01	18	0.01	20	0.03
$K_4 \times P_2$	15	499	0.15	198	0.21	172	0.21
$K_5 \times P_2$	1	9,350	5.55	1,252	6.28	866	3.84
$K_6 \times P_2$	0	192,360	220.95	4,151	109.50	1,861	42.53
$K_7 \times P_2$	0	3,674,573	8,433.96	10,635	739.53	2,440	169.81
$K_8 \times P_2$	0	-	-	23,048	3,301.41	2,553	351.32
$K_9 \times P_2$	0	-	-	-	-	2,561	715.21
$K_{10} \times P_2$	0	-	-	-	-	2,561	1,314.71

variables. The order in which the node variables are assigned is hard to predict, and it is difficult to see how the symmetry-breaking constraints can be tailored to be compatible with it. Because of the way that the two cliques are linked, it seems that early in the search labels are assigned in the second clique in a way that creates conflicts with the symmetry-breaking constraints when the corresponding nodes in the first clique are labelled later in search; hence, this leads the search into backtracking to try to resolve these conflicts. In the rest of the paper, other classes of graph containing multiple cliques are considered, but in these cases, the cliques are much less interlinked, and symmetry-breaking constraints do not encounter the same difficulties. Since symmetry-breaking constraints are easy to post, break all the symmetries in these problems, and offer the potential to prune variable domains early, they are a better choice than SBDS for these graphs. However, it is worth remembering that conflict with the variable ordering can sometimes cause them to do very poorly.

For  $n \geq 9$ , it appears from Table 1 that the size of the search tree explored by the edge-label model with SBDS no longer increases with  $n$ . This has also been observed in other classes of graph and will be discussed further in the next section. The runtime continues to increase with  $n$  because increasing the size of the problem means more variables and larger domain sizes, and hence constraint propagation is more time-consuming.

Cliques and nearly complete graphs with  $n$  nodes are graceful only for small values of  $n$ . Gallian lists a few other families of graphs involving cliques that are known not to be graceful; for instance, ‘windmill’ graphs  $K_n^{(m)}$  consisting of  $m$  copies of  $K_n$  with a common vertex are not graceful for  $m = 2$ . Intuitively, these graphs have too many edges compared to the number of nodes to be graceful, or else this eventually happens as  $n$  increases. In the light of this and the results in Table 1, it seems reasonable to conjecture that  $K_n \times P_2$  is not graceful for  $n > 5$ . In the next section, other graph families based on cliques will be considered which exhibit similar behaviour.

## 7 Sets of Overlapping Cliques

It is clearly possible to use constraint programming to show that some graphs are graceful and that some graphs are not. What would be required to show that a *class* of graph is not graceful (perhaps from some point onwards)?

If we consider cliques  $K_n$  and the proof given earlier that they are graceful only for  $n \leq 4$ , the proof demonstrates that when  $n \geq 6$ , it is impossible to construct an edge labelled  $q-5$ , where  $q$  is the number of edges, given that the edge labels  $q, q-1, \dots, q-4$  already exist. (The proof in the case  $n = 5$  is slightly shorter, since in that case the label  $q-4$  cannot be constructed.) If we applied this proof on an instance-by-instance basis, then when  $n = 6$ , it would be impossible to construct the edge labelled 11 ( $q=15$ ); when  $n = 7$ , it would be impossible to construct the edge labelled 16 ( $q = 21$ ); and so on. However, the proof for each instance would be essentially the same.

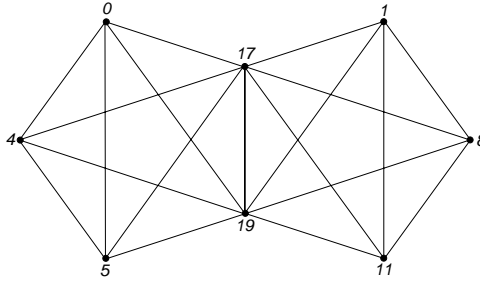
The search strategy used with the new model is based on the structure of Golomb's proof. This suggests that perhaps a proof that some class of graphs is not graceful from some point onwards could be constructed from the searches for individual instances that prove that each instance has no solution. For this to be possible, a first requirement is that the search is essentially the same for each instance, from some instance onwards on. And a first indication that the searches are essentially the same is that the number of backtracks is the same. In this section, this idea is explored, using a class of graphs that, like the  $K_n \times P_2$  graphs of the last section, contain multiple copies of a clique.

Gallian [6] summarises what is known about the gracefulness or otherwise of the class of graph denoted by  $B(n, r, m)$ , consisting of  $m$  copies of  $K_n$  with a  $K_r$  in common ( $n \geq r$ ). The case  $r = 1$  is the same as the windmill graphs  $K_n^{(m)}$ . For  $r > 1$ , the only graceful graphs in this class listed in Gallian's survey are:  $n = 3; r = 2; m \geq 1$ ;  $n = 4; r = 2; m \geq 1$ ;  $n = 4; r = 3; m \geq 1$ .

Using the edge-label model, it is easy to add to the list of graceful instances: for example, the graph  $B(5, 2, 2)$  is graceful, with a unique graceful labelling (except for symmetric equivalents) shown in Figure 4. It can also be shown that  $B(n, 2, 2)$  is not graceful for  $n = 6, 7, 8, 9$ , which according to Gallian's survey was not previously known. These results by themselves suggest that  $B(n, 2, 2)$  is not graceful for  $n \geq 6$ , and indeed, a result by Bermond and Farhi [2] shows that  $B(n, 2, 2)$  is not graceful for  $n \geq 9$ . Results for the model applied to these instances is shown in Table 2; the model is exactly as used for Table 1, except that only symmetry-breaking constraints are used and not SBDS.

It is notable that the number of backtracks is constant for  $n = 7, 8, 9$ ; this suggests that the same search tree is being explored for these instances. The number of assignments made during the search to each edge variable  $e_{q-i}$  is also the same in all three searches, making allowance for the fact that  $q$  is different in each case. If it can be shown the searches are in some sense equivalent for these three instances, and it can be assumed that they will continue to be equivalent for larger values of  $n$ , then a trace of the search could form the outline of a new proof, based on constraint programming, that these graphs are not graceful for  $n \geq 7$ .

In ILOG Solver, the user can define trace functions to execute after various events during the search, for instance when a value is about to be removed from the domain of a variable, when a value has just been removed from the domain of a variable, or when



**Fig. 4.** The graph  $B(5, 2, 2)$  with its unique graceful labelling (apart from symmetric equivalents).

**Table 2.** Search effort with the new model to find all graceful labellings of graphs  $B(n, 2, 2)$  or prove that there are none.

Graph	Solutions	backtracks	time (sec.)
$B(3, 2, 2)$	4	1	0.01
$B(4, 2, 2)$	4	22	0.01
$B(5, 2, 2)$	1	80	0.14
$B(6, 2, 2)$	0	116	1.72
$B(7, 2, 2)$	0	124	6.88
$B(8, 2, 2)$	0	124	15.26
$B(9, 2, 2)$	0	124	29.78

a variable has been set. These functions can be used to extract an instance-independent account of the search from the individual searches for the instances  $n = 7, 8, 9$ . The first step is to output any reference to an edge or node label  $> q/2$  in terms of  $q$  rather than a specific value. For instance, since  $q = 41, 55, 71$  when  $n = 7, 8, 9$ , the values 40, 54 and 70 respectively are all output as  $q - 1$ .

Most of the constraint propagation events occurring during the search are not relevant to its eventual failure. To tailor the trace output to try to produce a clear account of the search, the relevant events were defined.

For this class of graph, there are three symmetrically distinct sets of nodes: those in the first clique that are not in the common edge; those in the second clique that are not in the common edge; and the two nodes in the common edge. (Although the two cliques are indistinguishable in the graph, they are distinguished in the CSP by the symmetry-breaking constraints.) The three sets of nodes are denoted by  $K_n^a$ ,  $K_n^b$  and  $K_2$  respectively in the trace output.

The only relevant events affecting the node label variable  $n_i$  are: the node label must be used (because it is required to form an edge label); it cannot be used (because it would create a duplicate edge label); it must or must not be assigned to a node in one of the three sets (denoted for instance as  $n_i \in K_n^a$  in the trace output).

The only events relating to edge labels that are output are those in which the value of an edge-label variable is set; this is output as  $e_i = (j, j+i)$ . This can happen either as the result of a search choice, or because two node labels have been assigned to adjacent

nodes and created this edge label, or because all other values have been removed from the domain of the variable. The last event is the most interesting because it can lead to further constraint propagation. A value  $j$  is removed from the domain of an edge variable  $e_i$  either because one or both of the node labels  $j$  and  $j + i$  cannot be used; or because one of the node labels is already assigned to a node in  $K_n^a$  or  $K_n^b$  and the other label cannot be assigned to an adjacent node; or because one node label is already assigned to a node in  $K_n^a$  and the other to a node in  $K_n^b$ .

As well as using the Solver trace facilities to output details of the relevant constraint propagation events, a summary of the state of the search is output whenever the variable ordering heuristic chooses the next variable to assign. The variable assignments made so far are listed, as well as any node label variables that must be used or cannot be in one of the sets of nodes. The search follows the Solver default of making binary choices, either assigning a value to a variable ( $var = val$ ) or not assigning that value to that variable ( $var \neq val$ ). On each branch, the relevant choice defining the branch is output.

The trace information is output as Latex; a small example is shown in Table 3. This fragment was obtained by solving the subproblem created by labelling the nodes in the common edge 0 and  $q$  and insisting that the edge labelled  $q-2$  joins nodes labelled 2 and  $q$  (in the instance  $n = 7$ ). It takes only three backtracks to prove that the subproblem has no solution, compared to 124 for the complete problem. The original output has been further reduced by about half by removing all the output relating to propagation between successive choice point on the same branch: the summary of the state of the search when each choice is made presents the same information more clearly. This leaves the output relating to the constraint propagation on each branch between the last choice point and the point where the search backtracks. For clarity, a few propagation events that were not relevant to the failure have also been removed and a brief explanation of each failure (in italics) is given; these last two changes have obviously required manual intervention, but have been done in order to explore what would be needed to explain the failure reasonably comprehensibly.

In the complete search traces for the three instances, the choices made and so the search tree explored are identical (apart from the value of  $q$ ). The smallest edge label  $> q/2$  whose value is set during the search is  $q - 20$  and the largest edge label  $< q/2$  is 12; the smallest edge label set at a choice point is  $q - 9$ , but smaller edge labels are set by constraint propagation. The traces do differ very slightly from instance to instance because events can be added to the propagation queue in a different order.

The trace given in Table 3 does not unfortunately give as clear an explanation of each failure as would be desirable. One difficulty is that the list of events preceding a failure does not necessarily appear in a logical order from the point of view of explaining it. Nevertheless, the information that is output before each failure is sufficient to be able to understand (with a little effort) why the previous choice, given the state of the search at that point, led to a failure.

It is worth noting that the proof given by Beutner and Harborth [3] that  $K_n - e$  is graceful only if  $n \leq 5$  is based on their search procedure but is given in a very compressed format (which reduces the proof given in section 4 to just five lines). The proof gives the structure of the search tree, with the choices available at each choice point. It lists the assignments made to node labels during the search (whether by choice

**Table 3.** Search to demonstrate that there is no labelling of  $B(n, 2, 2)$  with the common edge labelled 0 and  $q$  and the edge label  $q - 2$  formed from node labels 2 and  $q$ .

Edge labels set:  $e_1 = (0, 1)$ ;  $e_2 = (0, 2)$ ;  $e_3 = (q - 3, q)$ ;

$e_{q-3} = (0, q - 3)$ ;  $e_{q-2} = (2, q)$ ;  $e_{q-1} = (1, q)$ ;  $e_q = (0, q)$ ;

Node labels set:  $n_0 \in K_2$ ;  $n_1 \in K_n^a$ ;  $n_2 \in K_n^b$ ;  $n_q \in K_2$ ;

Node labels partly set:  $n_4 \notin K_n^b$ ;  $n_5 \notin K_n^b$ ;

Node labels not used: 3; 4;  $q - 2$ ;  $q - 1$ ;

Choose  $n_{q-3} \in K_n^a$

Edge labels set:  $e_1 = (0, 1)$ ;  $e_2 = (0, 2)$ ;  $e_3 = (q - 3, q)$ ;

$e_{q-4} = (1, q - 3)$ ;  $e_{q-3} = (0, q - 3)$ ;  $e_{q-2} = (2, q)$ ;  $e_{q-1} = (1, q)$ ;  $e_q = (0, q)$ ;

Node labels set:  $n_0 \in K_2$ ;  $n_1 \in K_n^a$ ;  $n_2 \in K_n^b$ ;  $n_{q-3} \in K_n^a$ ;  $n_q \in K_2$ ;

Node labels partly set:  $n_5 \notin K_n^b$ ;  $n_{q-6} \notin K_n^a$ ;  $n_{q-5} \notin K_n^a$

Node labels not used: 3; 4;  $q - 4$ ;  $q - 2$ ;  $q - 1$ ;

Choose  $e_{q-5} = (0, q - 5)$

$n_5 \notin K_n^a$ ;  $n_5$  is not used

$n_{q-5}$  must be used;  $n_{q-5} \in K_n^b$

$e_{q-7} = (2, q - 5)$

$e_5 = (q - 5, q)$

$n_{q-6}$  is not used

$n_{q-8}$  is not used

$n_{q-7}$  is not used

$e_{q-6} = (6, q)$

$n_7$  is not used

$n_6 \notin K_n^a$

$e_{q-8} = (8, q)$

$n_8$  must be used;  $n_8 \in K_n^a$

$n_6$  must be used;  $n_6 \in K_n^b$

$e_{q-9} = (0, q - 9)$

$n_{q-9}$  must be used

$e_6 = (0, 6)$

$e_4 = (2, 6)$

*fail:  $n_{q-9} \in K_n^a$  gives a 2nd edge labelled 6 from  $(q - 9, q - 3)$*

*$n_{q-9} \in K_n^b$  gives a 2nd edge labelled 4 from  $(q - 9, q - 5)$   $\diamond$*

Choose  $e_{q-5} \neq (0, q - 5)$

$e_{q-5} = (5, q)$

$n_5$  must be used;  $n_5 \in K_n^a$

$n_{q-5}$  is not used

$e_4 = (1, 5)$

$n_6$  is not used

$n_{q-6}$  is not used

$e_{q-6} = (0, q - 6)$

*fail:  $q - 6$  is not used  $\diamond$*

Choose  $n_{q-3} \notin K_n^a$

$n_{q-3} \in K_n^b$

$e_{q-5} = (2, q - 3)$

$n_{q-4}$  must be used;  $n_{q-4} \in K_n^a$

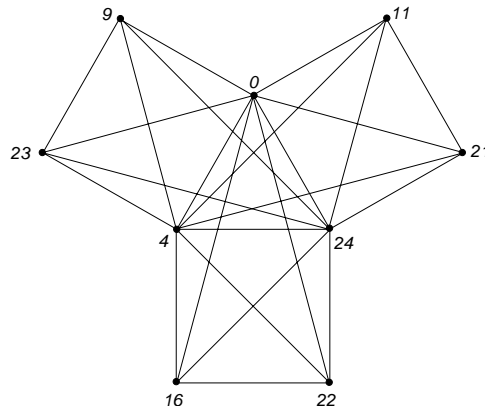
*fail:  $(q - 4, 1)$  gives a 2nd edge labelled  $q - 5$   $\diamond$*

or as the result of another assignment) and the largest edge label not yet used. It differs significantly from Table 3 in that when the next largest edge label cannot be constructed, given the choices already made, it is simply recorded that the search backtracks at that point, without any further explanation. Although it is possible to reconstruct the reason for the failure from the previous choices, it requires significant work. Their proof, based on a search with 13 backtracks, occupies just 27 lines. Using the same format, Table 3 would occupy about six lines, and most of the trace information would be lost.

Bundy [4] discusses the acceptability (or rather the unacceptability) of large computer-generated proofs in mathematics. A proof that  $B(n, 2, 2)$  is not graceful for  $n > 7$  based on a search which requires 124 backtracks and in the style of Table 3 would occupy about 50 pages. However, cutting out explanatory detail does not seem to be an adequate answer. Bundy suggests breaking a long automated proof into small lemmas, where possible. In this case, we could think of each failure during the search as requiring a lemma that the choices made along this branch of the search tree cannot lead to a graceful labelling. Hence, Table 3 could be rewritten as three lemmas, one for each failure, embedded within a structure reflecting the search tree.

## 8 Larger classes

The edge-label model has been used to investigate other graphs in the general  $B(n, r, m)$  class. The graph  $B(4, 3, 2)$ , i.e. two copies of  $K_4$  with a common triangle, was already known to be graceful; the result in [2] shows that  $B(n, 3, m)$  is not graceful for  $n \geq 11$  if  $m \geq 2$ . With the edge-label model, it can be shown that  $B(n, 3, 2)$  is also not graceful for  $n = 6, 7, 8, 9, 10$ . The search to prove this for the last three instances takes 143 backtracks in each case, suggesting that, as before, these searches are essentially identical.  $B(4, 3, 3)$  was already known to be graceful.  $B(5, 3, 3)$  is also graceful, with 5 possible distinct labellings, of which one is shown in Figure 5; this is a new result. For larger



**Fig. 5.** The graph  $B(5, 3, 3)$  with one of its graceful labellings.

values of  $n$ ,  $B(n, 3, 3)$  is not graceful for  $n = 6$  and  $7$ . The search effort to prove that an instance has no graceful labelling is still increasing rapidly with  $n$  at these problem sizes: for  $n = 7$ , the search takes 5,944 backtracks. One might expect that the search effort may exhibit similar behaviour to that shown in Table 2, i.e. that at some point the search effort reaches a plateau and appears not to increase further. However, even if an acceptable proof can be based on a search which requires 100 or so backtracks, it seems doubtful that this can be done for a search requiring thousands.

## 9 Conclusions

A CSP model for finding a graceful labelling of a graph or proving that it is not graceful has been presented, based on constructing the edge labels in descending order. This model is far more efficient than the previous model based on assigning a label to each node. This demonstrates again the importance of having a good model of the problem to be solved, and also that what seems a natural CSP model may not prove to be a good model. The new model in this case was developed not by reformulating the existing model, but by adopting an approach from the graph theory literature [8]; non-CP approaches to solving a problem may be a good source of modelling ideas in other cases.

The search strategy used with the edge-label model has already been used by Beutner and Harborth [3] in a special-purpose algorithm for finding graceful labellings. The main advantage of using this strategy in a constraint programming framework is the constraint propagation it provides; in [3], checking that a new node label does not create duplicate edge labels is only done after the node label is assigned, whereas in solving the CSP, it can be determined in advance that the node label is not consistent with the assignments already made. Furthermore, a trace of the constraint propagation provides reasons for a branch of the search to fail and thus can be used to explain the search; as discussed in section 3 it may be possible in some cases to use trace information as the basis for a proof that a class of graphs is ungraceful.

The edge-label model has so far been applied to classes of graph which contain multiple copies of the clique  $K_n$ . Some new results have been obtained; graceful labellings have been found for  $B(5, 2, 2)$  and  $B(5, 3, 3)$ , which were not previously known to be graceful, and it has been shown for the first time that several instances of the same classes are not graceful.

Future work will consider further how to construct a proof, that the graphs  $B(n, 2, 2)$  and similar graphs are not graceful beyond a certain size, from the searches to prove that CSP instances have no solution. Although Bermond and Farhi [2] already dealt with the class  $B(n, r, m)$ , there are related graphs, such as those consisting of two unequal-sized cliques with a common edge, that are not covered by their result, but that could be amenable to a search-based proof. Currently, we are investigating a single CSP that is a relaxation of every CSP instance representing the graceful labelling of  $B(n, 2, 2)$  with  $n \geq 7$ , and also covers the case where the cliques are not the same size. If the relaxation has no solution, then none of the instances has a solution either. This work will be described in a later paper.

The conjecture that graphs  $K_n \times P_2$  are not graceful for  $n > 5$  seems harder to prove. A key point in presenting the search trace given in Table 3 is that in  $B(n, r, m)$

graphs, the nodes fall into three sets. The nodes within each set are indistinguishable, and this limits the number of different search events affecting each node label that need be considered. In  $K_n \times P_2$  graphs, although all the nodes are initially indistinguishable, they become distinct as nodes are labelled and other nodes have different relationships to the labelled nodes; hence the number of different cases to consider is far greater. Nevertheless, the fact that the search tree appears not to increase in size for  $n \geq 9$  suggests that in theory a proof might be constructed from the search.

Other classes of graph constructed from multiple copies of cliques will also be considered; from the evidence reported in this paper, it seems likely that these will also be graceful only for small instances. Further elucidation of this type of graph would be a useful contribution to the study of graceful graphs.

## Acknowledgments

This material is based in part on works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075. I am grateful to Ian Gent for helpful comments. Thanks are also due to colleagues at 4C, and especially to Nic Wilson.

## References

1. N. Beldiceanu. Global constraints as graph properties on structured network of elementary constraints of the same type. Technical Report Technical Report T2000/01, SICS, 2000.
2. J. C. Bermond and G. Farhi. Sur un problème combinatoire d'antennes en radioastronomie II. *Annals of Discrete Mathematics*, 12:49–53, 1982.
3. D. Beutner and H. Harborth. Graceful labelings of nearly complete graphs. *Results in Mathematics*, 41:34–39, 2002.
4. A. Bundy. A Very Mathematical Dilemma. *Computer Journal*, 2006. (In press).
5. J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-Breaking Predicates for Search Problems. In *Proceedings KR'96*, pages 149–159, Nov. 1996.
6. J. A. Gallian. A Dynamic Survey of Graph Labeling. *The Electronic Journal of Combinatorics*, (DS6), 2005. 9th edition.
7. I. P. Gent and B. M. Smith. Symmetry Breaking During Search in Constraint Programming. In W. Horn, editor, *Proceedings ECAI'2000, the European Conference on Artificial Intelligence*, pages 599–603, 2000.
8. S. W. Golomb. How to number a graph. In R. C. Read, editor, *Graph Theory and Computing*, pages 23–37. Academic Press, 1972.
9. Y. C. Law and J. H. M. Lee. Symmetry Breaking Constraints for Value Symmetries in Constraint Satisfaction. *Constraints*, 11:221–267, 2006.
10. I. J. Lustig and J.-F. Puget. Program Does Not Equal Program: Constraint Programming and Its Relationship to Mathematical Programming. *INTERFACES*, 31(6):29–53, 2001.
11. K. E. Petrie and B. M. Smith. Symmetry Breaking in Graceful Graphs. Technical Report APES-56-2003, APES Research Group, 2003. Available from <http://www.dcs.st-and.ac.uk/~apes/apesreports.html>.
12. J.-F. Puget. Breaking symmetries in all different problems. In *Proceedings of IJCAI'05*, pages 272–277, 2005.
13. B. Smith and K. Petrie. Graceful Graphs: Results from Constraint Programming. <http://4c.ucc.ie/~bms/Graceful/>, 2003.